CE222046 – PSoC 6 BLE Throughput Measurement

Objective

This code example demonstrates how to maximize the BLE throughput on PSoC® 6 MCU with Bluetooth Low Energy (BLE) Connectivity device.

Overview

This example uses the BLE GATT layer notification to achieve maximum throughput between a GATT server and a GATT client. The example has two projects:

1. CE222046_GATT_Out (GATT server): Data transfer using GATT Notification on a Characteristic – Outgoing data.
2. CE222046_GATT_In (GATT client): Data transfer using GATT Notification on a Characteristic – Incoming data.

The GATT server device initializes a buffer of 495 bytes. Once the connection is established between the GATT client and GATT server, the GATT client device enables the notification on the Server. The Server then keeps sending data continuously. The BLE data throughput is calculated by the Client device and displayed on a UART terminal emulator.

Requirements

Tool: PSoC Creator™ 4.2

Programming Language: C (Arm® GCC 5.4.1)

Associated Parts: All PSoC 6 MCUs with BLE Connectivity

Related Hardware: CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

Design

There are two projects for GATT Notifications: CE222046_GATT_In and CE222046_GATT_Out. The CE222046_GATT_In serves the role of GATT client and a GAP Central device. The CE222046_GATT_Out project serves as a GATT server and GAP Peripheral.

The GATT client establishes a connection with the GATT server and enables the notification on the Server side. The Server uses the GATT notification on a custom Characteristic to send data to the GATT client device continuously. The GATT client receives the data sent by the Server and calculates the amount of data received in 10 seconds. The calculated throughput is displayed on a UART terminal emulator once is every 10 seconds.

Figure 1 (a) and (b) shows the project schematic for CE222046_GATT_In and CE222046_GATT_Out projects respectively.

Refer to Parameter Settings section for details about the configuration settings of the components used in the two projects.

Figure 1 (a). PSoC Creator Project Schematic: CE222046_GATT_In

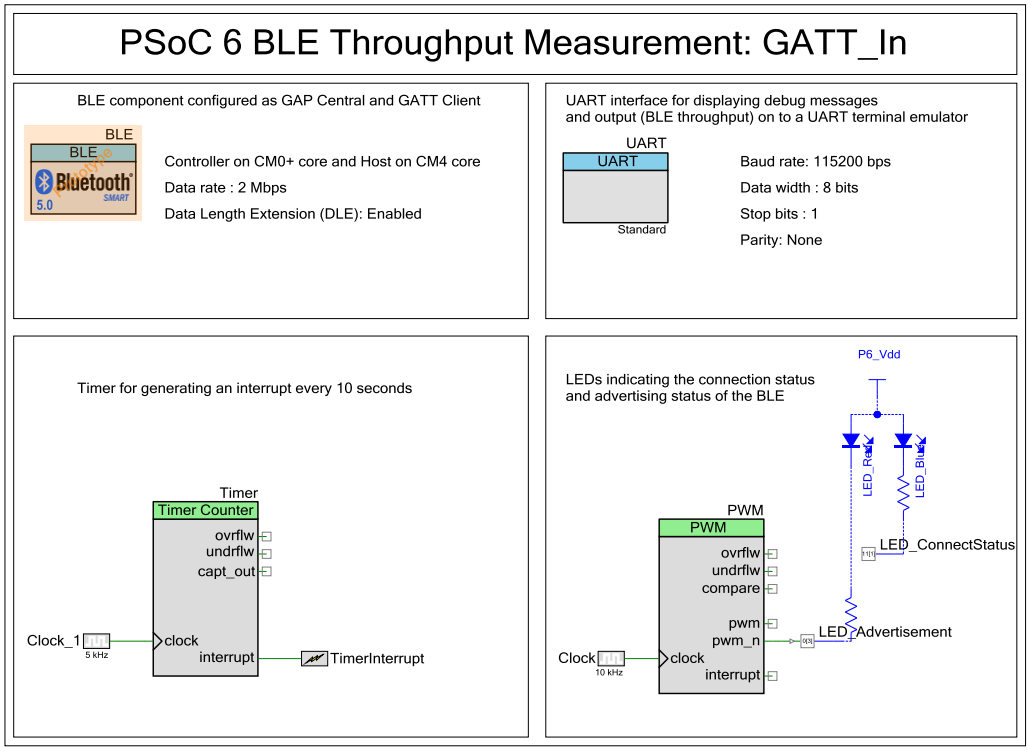
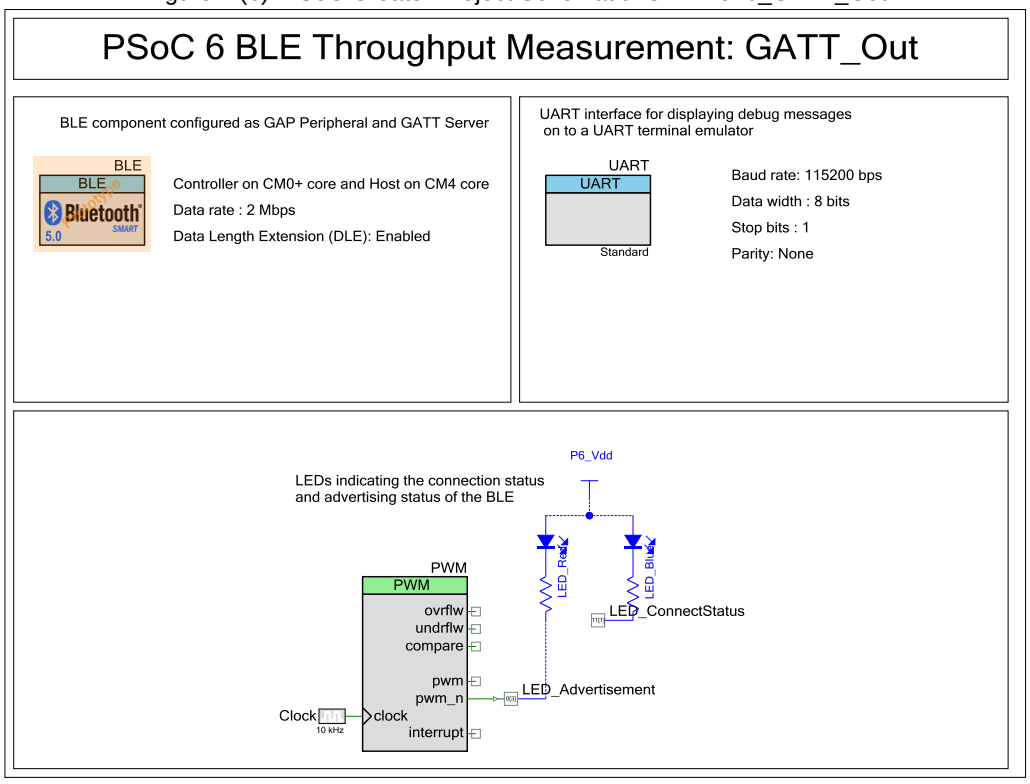


Figure 2 (b). PSoC Creator Project Schematic: CE222046_GATT_Out



Hardware Setup

Use the CY8CKIT-062-BLE PSoC 6 Pioneer Kit to program and test this code example. Two kits are required to test this code example: one each for CE222046_GATT_In project and CE222046_GATT_Out project. The kit is used in its default configuration. If you have modified the hardware configuration, see the Pioneer Kit guide for the steps to revert to the default configuration.

Software Setup

This project uses Tera Term as a UART terminal emulator for displaying the measured BLE Throughput. You can use any other serial terminal emulator as well for this project.

Operation

1. Connect two Pioneer Boards to your PC using the provided USB cable through the USB connector (J10).
2. Build the projects CE222046_GATT_In and CE222046_GATT_Out and program them into the PSoC 6 MCU on two different kits. Choose **Debug** > **Program**. For more information on device programming, see Pioneer Kit guide. Flash for both CPUs is programmed in a single program operation.

   Note: During the build process, do not replace *stdio_user.h* and *stdio_user.c* file if prompted by PSoC Creator.
3. Open two instances of UART terminal emulator. This example uses Tera Term as the UART terminal. Set the baud rate to 115200 bps. Use the default settings for other serial port parameters. Connect the KitProg2's COM port of the two kits to the UART terminals: one for CE222046_GATT_In project and other for CE222046_GATT_Out project.
4. Press the reset switch (SW1) on the kit with CE222046_GATT_In project programmed. The scanning for the Peripheral is indicated by blinking of red LED on the Kit. The status of scanning is also displayed on the UART terminal.
5. Similarly, press the reset switch (SW1) on the kit with CE222046_GATT_Out project programmed. The advertisement status by the Peripheral is indicated by blinking of red LED on the Kit. The status of advertisement is also displayed on the UART terminal.
6. The GATT client connects automatically to the GATT server device. Once they are connected, the blue LEDs on both the kits turn ON indicating that the connection has been established successfully.
7. The measured BLE throughput is then displayed on the UART terminal connected to the CE222046_GATT_In project. The measured BLE throughput is updated every 10 seconds.

   Figure 3 and Figure 4 shows the sample output screenshots.

Figure 3. Output Screenshot of CE222046_GATT_Out Project

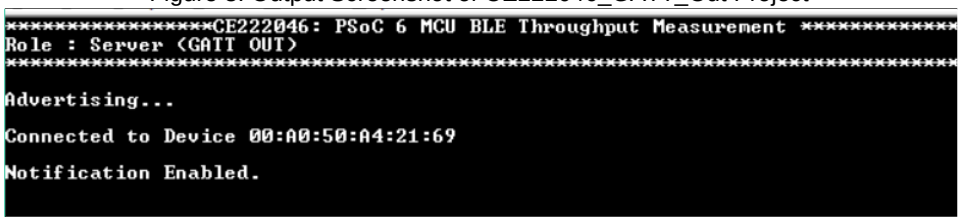
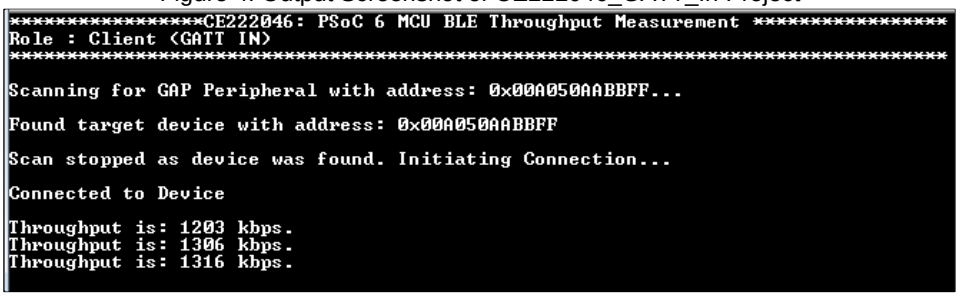


Figure 4. Output Screenshot of CE222046_GATT_In Project



The sections that follow discuss the Components, parameter settings, and resources used to make the example.

Components

Table 1 lists the PSoC Creator Components used in this example (both common and specific for each project), how they are used in the design, and the non-default settings required so they function as intended.

Table 1. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
BLE	BLE	BLE Communication	Refer to the Parameter Settings section for other parameters
UART	UART	Print string messages on a terminal window	**Baud Rate (bps)**: 115200 **Tx/Rx Mode:** Tx only
Digital Output Pin	LED_ConnectStatus	Drive blue RGB LED	**HW connection**: Uncheck
Digital Output Pin	LED_Advertisement	Drive red RGB LED	Default setting only
TCPWM	PWM	Generate a signal with 50% duty cycle and with a period of 1 second.	**Period 0**: 9999 **Compare 0**: 5000
TCPWM (used in CE222046_GATT_In project)	Timer	Generate an interrupt every 10 seconds	**Period**: 50000 **Interrupt Source**: Overflow/Underflow
Interrupt (used in CE222046_GATT_In project)	TimerInterrupt	Set the core and priority for the interrupt	Default setting only

Parameter Settings

This section discusses the non-default configuration settings used by the BLE component in CE222046_GATT_In and CE222046_GATT_Out projects.

BLE Component settings in CE222046_GATT_In

CE222046_GATT_In is configured as GAP Central device with BLE controller run by the CM0+ core and Host on the CM4 core. It acts a GATT client with a custom service: Throughput Custom Service. It uses an MTU size of 512 bytes in order to maximize the BLE throughput.

In the Link Layer Settings, Data Length Extension (DLE) is enabled by setting TX and RX payload size to 251 bytes. The data rate is set to 2 Mbps. Figure 5 through Figure 8 show the configuration settings used for CE222046_GATT_In project.

Figure 5. CE222046_GATT_In: General Settings

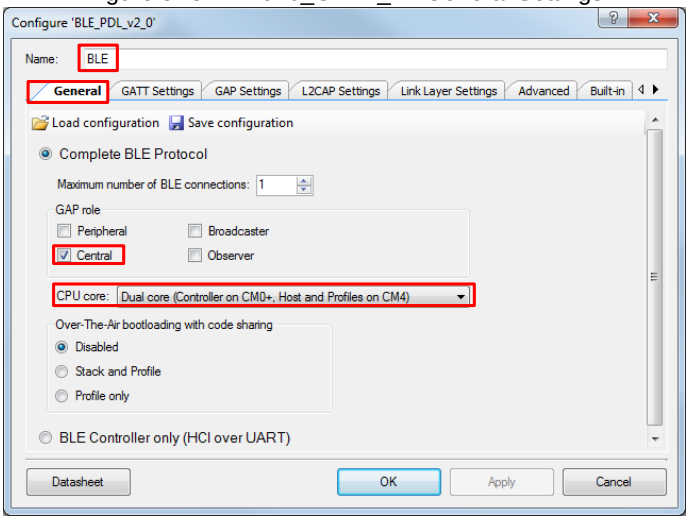


Figure 6. CE222046_GATT_In: GATT Settings

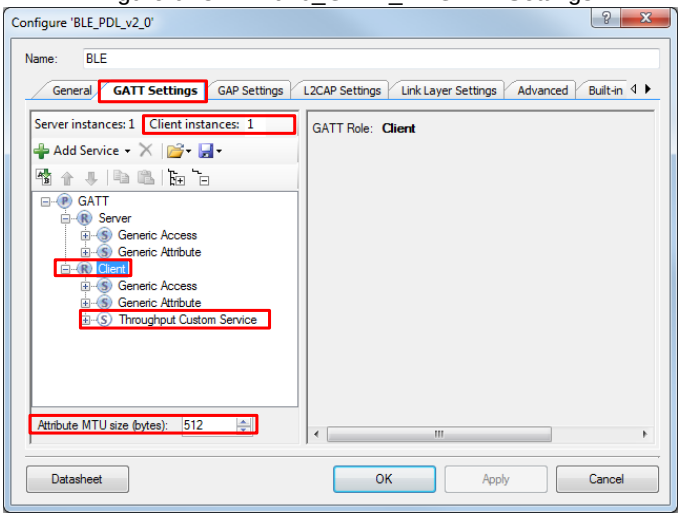


Figure 7. CE222046_GATT_In: GAP Settings

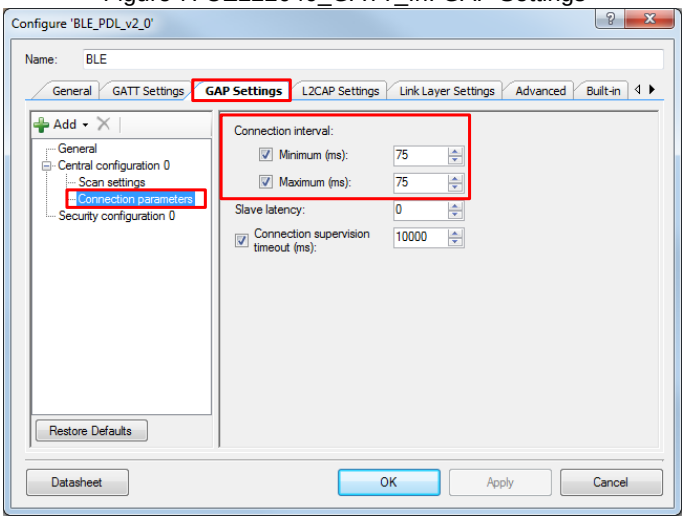
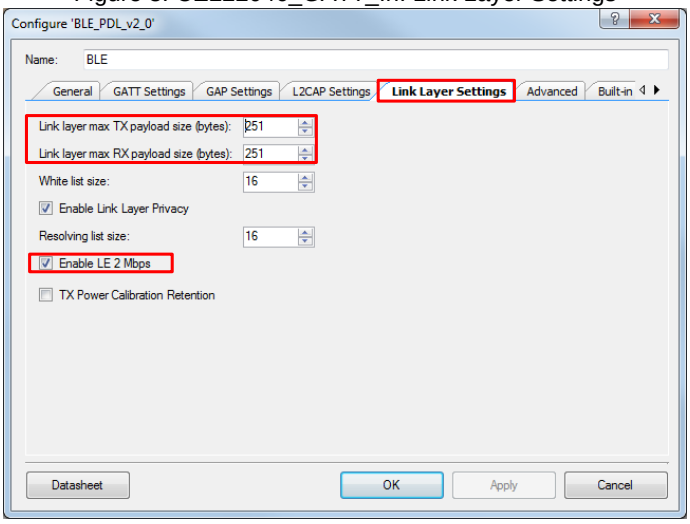


Figure 8. CE222046_GATT_In: Link Layer Settings



Under the GAP Settings tab, in the General parameter section enter the Device name as “GATT_In”.

In the GAP Settings tab, the connection interval is set to 75 ms. This value is used so that BLE bandwidth is used effectively. Each notification packet is of size 495 bytes, which takes a processing time of 2.34 ms (refer to BLE Component datasheet for details). Therefore, in a connection interval of 75 ms, 32 notification packets (75 ms / 2.34 ms) are sent and only a nominal 0.05 packet is lost.

BLE Component settings in CE222046_GATT_Out

CE222046_GATT_Out is configured as GAP Peripheral device with BLE Controller run by the CM0+ core and Host on the CM4 core. It acts a GATT server with a custom service: BLE Throughput. It uses an MTU size of 512 bytes in order to maximize the BLE throughput.

In the Link Layer Settings, Data Length Extension (DLE) is enabled by setting TX and RX payload size to 251 bytes. The data rate is set to 2 Mbps. Figure 9 through Figure 14 show the configuration settings used for CE222046_GATT_Out project.

Figure 9. CE222046_GATT_Out: General Settings

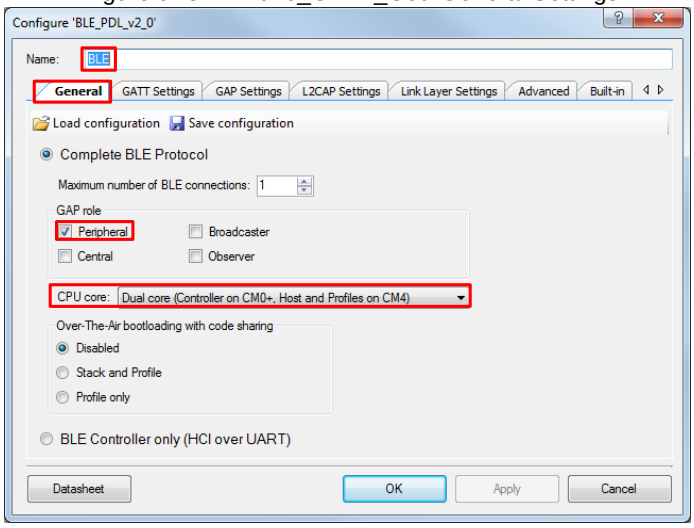


Figure 10. CE222046_GATT_Out: GATT Settings

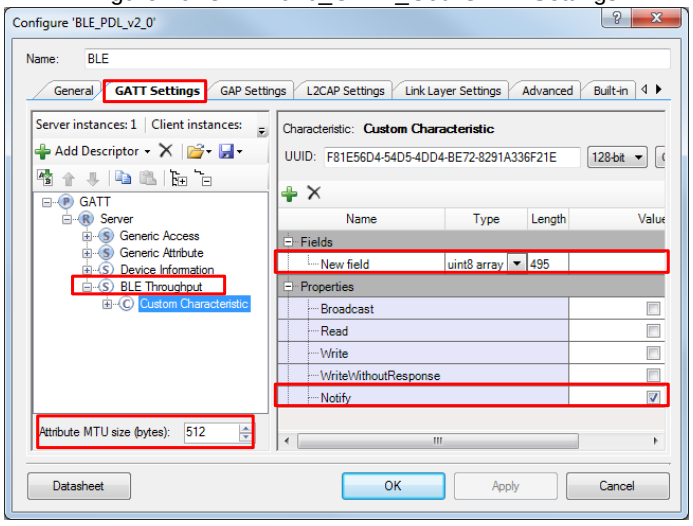


Figure 11. CE222046_GATT_Out: GAP Settings

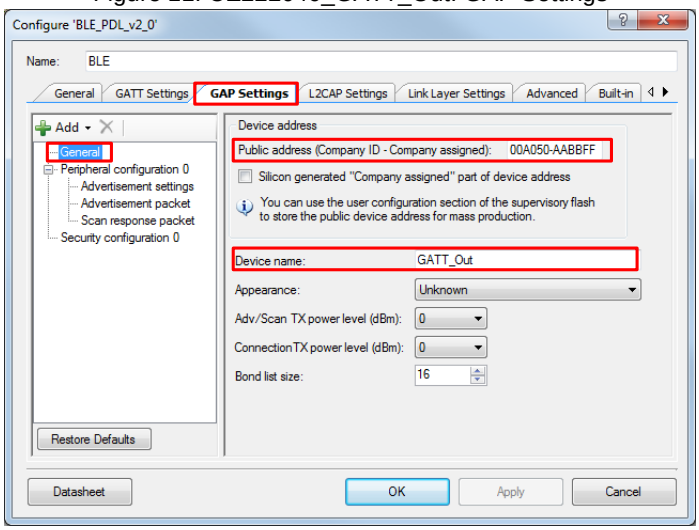


Figure 12. CE222046_GATT_Out: Link Layer Settings

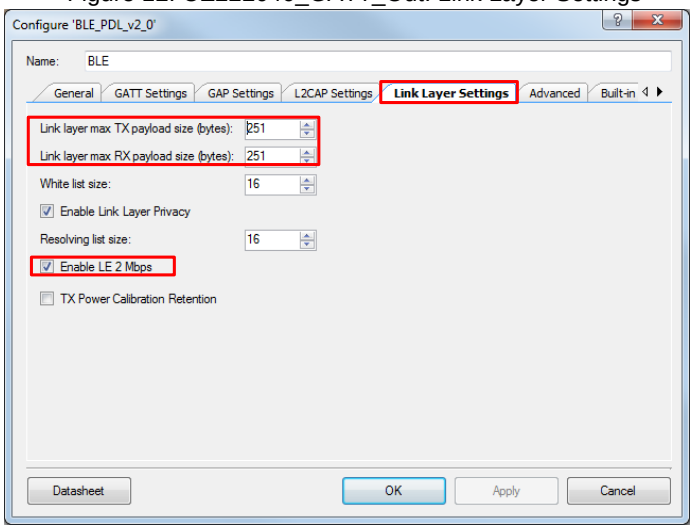


Figure 13. CE222046_GATT_Out: GAP Advertisement settings

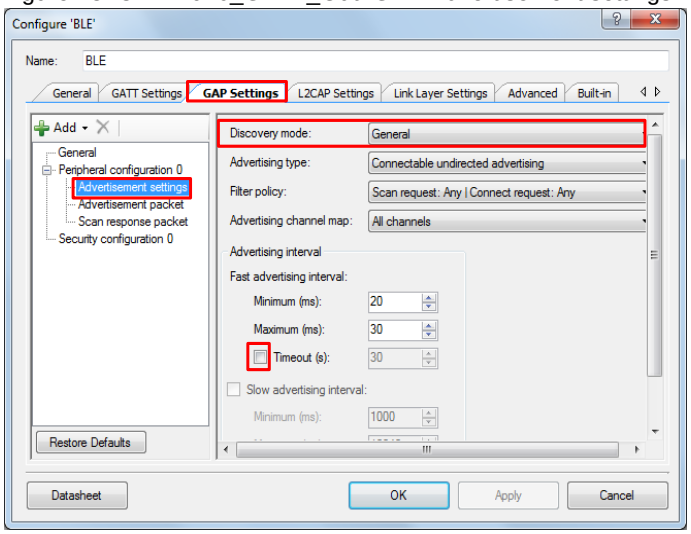
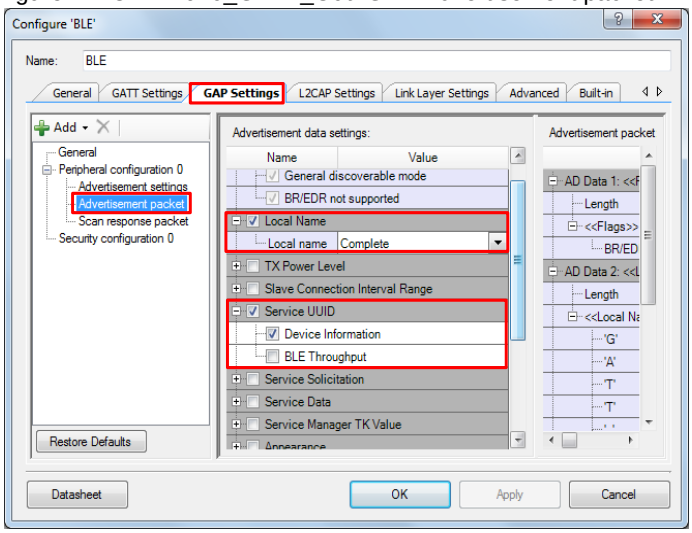


Figure 14. CE222046_GATT_Out: GAP Advertisement packet



The CE222046_GATT_Out device advertises with a public address of 00A050-AABBFF. The CE222046_GATT_In device scans for a GAP Peripheral (CE222046_GATT_Out) with this address and connects to it automatically.

Design-Wide Resources

The projects in this example uses the KitProg2 USB-UART bridge to communicate with UART terminal emulator running on your PC. PSoC 6 Pioneer Kit uses the pin **P5[1]** as UART TX. Figure 15 shows the pins used in both CE222046_GATT_In and CE222046_GATT_Out projects.

Figure 15. Device Pin Assignments

Name	Port	Pin	Lock
\UART:tx\	P5[1]	K6	☑
LED_Advertisement	P0[3]	E3	☑
LED_ConnectStatus	P11[1]	E5	☑

Figure 16 and Figure 17 show the system interrupt configuration used in this example.

Figure 16. Interrupt Configuration for CE222046_GATT_In Project

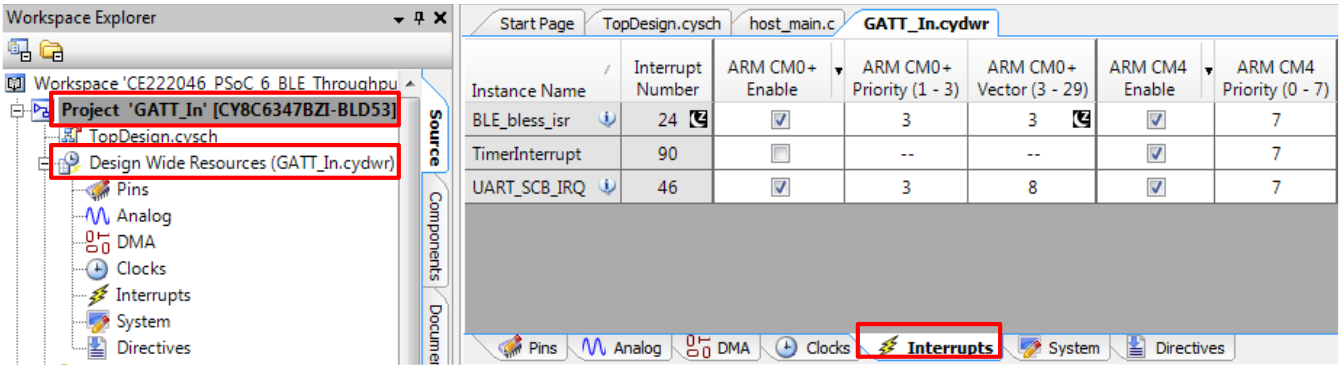
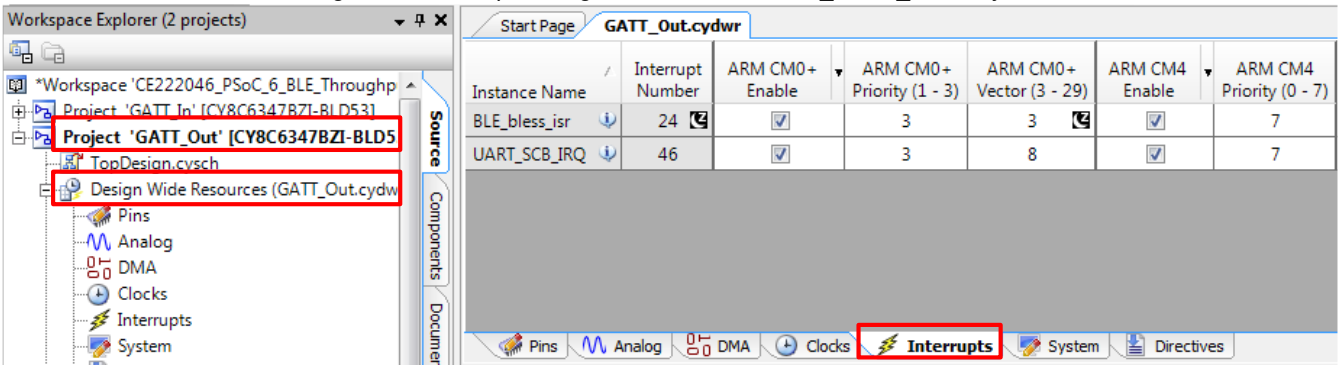


Figure 17. Interrupt Configuration for CE222046_GATT_Out Project



Related Documents

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes the PSoC 6 MCU with BLE Connectivity device family.
PSoC Creator Component Datasheets	
Bluetooth Low Energy	Facilitates designing applications requiring BLE connectivity
UART	Provides asynchronous serial communications
Pins	Supports connection of hardware resources to physical pins
Interrupt	Provides an interface to connect hardware signals to a CPU interrupt request line.
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheet	PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual
Development Kit (DVK) Documentation	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	

Document History

Document Title: CE222046 - PSoC 6 BLE Throughput Measurement

Document Number: 002-22046

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6005530	VKVK	12/26/2017	New code example
*A	6074748	VKVK	02/19/2018	Updated the project with PSoC Creator 4.2

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.